**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**"Уфимский университет науки и технологий"**

**Кафедра** Высокопроизводительных вычислений и дифференциальных уравнений

**Дисциплина:** Математическое моделирование.

**Отчет по лабораторной работе № 3**

**Тема:** «Моделирование распространения электромагнитных волн в плоском волноводе»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМ-457 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Акмурзин М.Э. |  |  |  |
| Принял | Лукащук С.Ю. |  |  |  |

**Уфа 2025**

**Цель работы:** получить навык моделирования распространения электромагнитных волн в неоднородных волноводах на основе решения краевой задачи для уравнения Гельмгольца

**Задание**

Рассматривается задача распространения электромагнитной волны в плоском волноводе.

В случае полуограниченного однородного канала постоянного сечения математическая модель имеет вид краевой задачи для уравнений Гельмгольца. Рассматриваются граничные условия первого рода. Модельная задача имеет следующий вид:

|  |  |
| --- | --- |
|  | (1) |

где – потенциал электромагнитного поля,  – волновое число.

В случае, когда канал имеет поперечную неоднородность свойств, уравнение модифицируется:

где функция описывает вид неоднородности. В данной работе будут рассматриваться каналы со слабой неоднородностью, то есть каналы, для которых справедливо представление , где – малый параметр.

Таким образом, рассматривается краевая задача

|  |  |
| --- | --- |
|  | (2) |

Функция задается в виде -функции:

|  |  |
| --- | --- |
|  | (3) |

Для модели (2), (3) реализовать следующее:

1. разработать конечно-разностную схему численного решения задачи (2), (3) с аппроксимацией -функции на сетке с применением PML-слоя для моделирования условия при ;
2. реализовать разработанную схему в виде вычислительной программы;
3. для заданной функции (в соответствии с вариантом) провести серию вычислительных экспериментов, направленных на исследование влияния на решение следующих параметров задачи:

Вариант 2:

**Ход работы**

Дискретизируем систему. Для начала введем сетку:

Обозначим:

Заменим производные, численная схема примет вид:

Перепишем уравнение в следующем виде:

Получим систему размером уравнений.

**PML-слой**

Длина PML-слоя . Заменим

где – функция затухания.

Тогда

Представим функцию u в следующем виде

Уравнение Гельмгольца перепишется в следующем виде

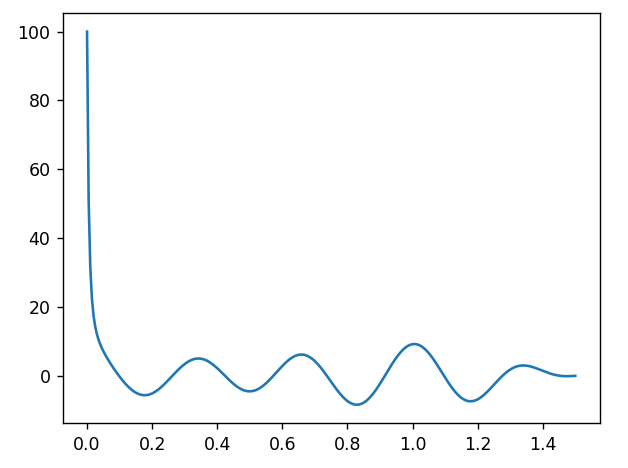
которое можно расщепить на действительную и комплексную части

Условие стыковки

Дискретизируем систему

Условия стыковки

Проведем серию вычислительных экспериментов и исследуем влияние параметров на решение задачи. Для анализа влияния параметров на результат вычислений, будем поочередно варьировать один из параметров, фиксируя все остальные.



**Параметр**

Зафиксируем

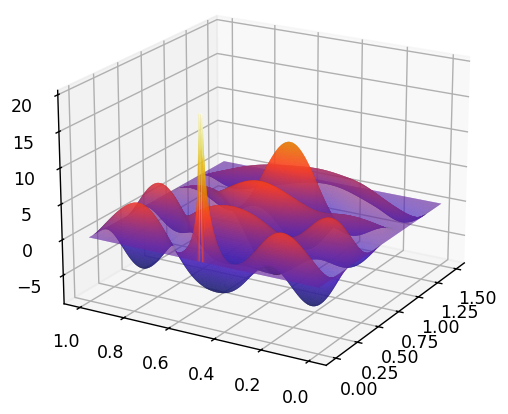


Рисунок 1 –

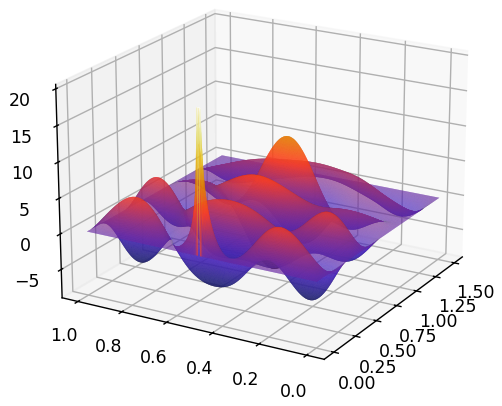


Рисунок 2 –

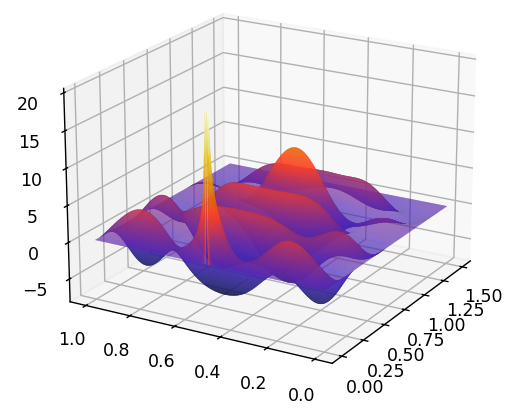


Рисунок 3 –

**Параметр**

Зафиксируем

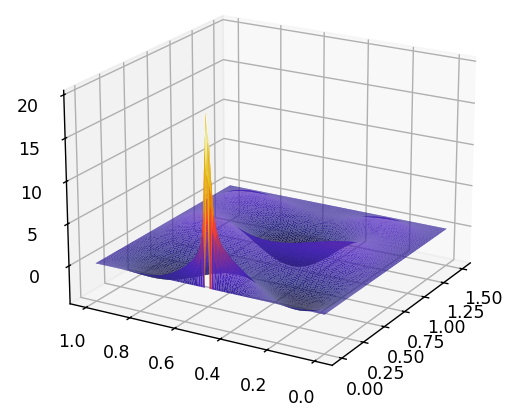


Рисунок 4 –

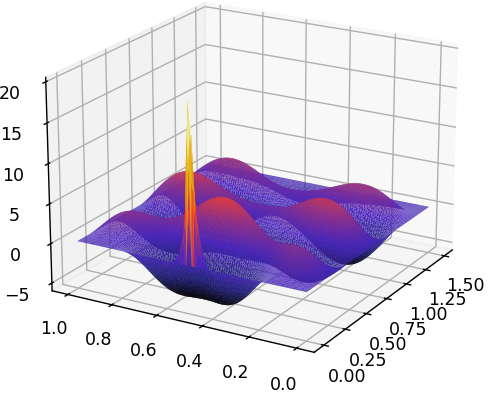


Рисунок 5 –

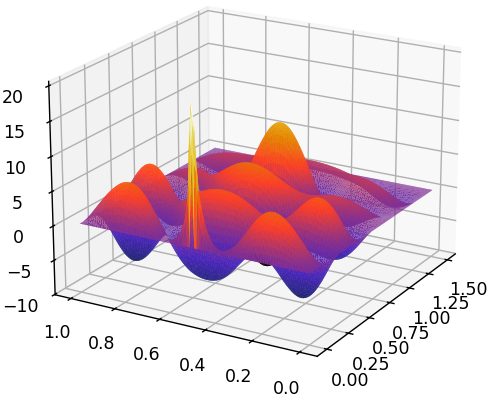


Рисунок 6 –

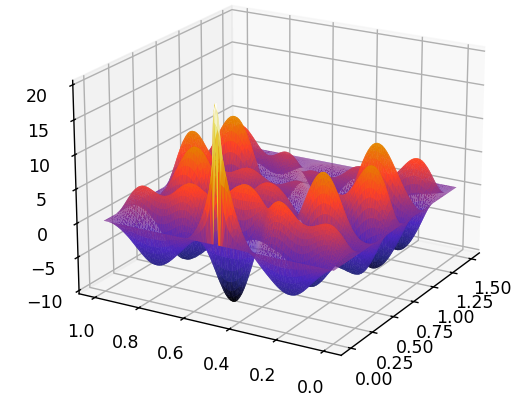


Рисунок 7 –

**Параметр**

Зафиксируем

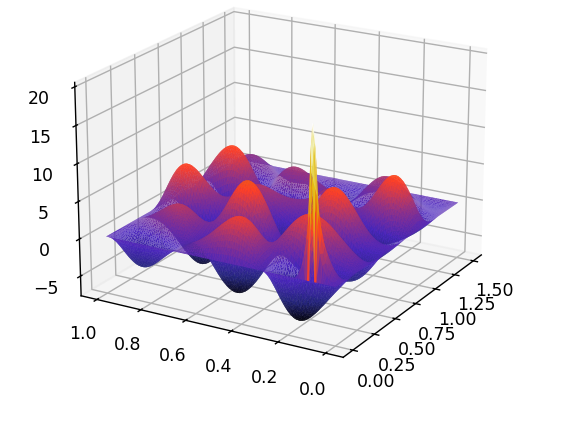


Рисунок 8 –

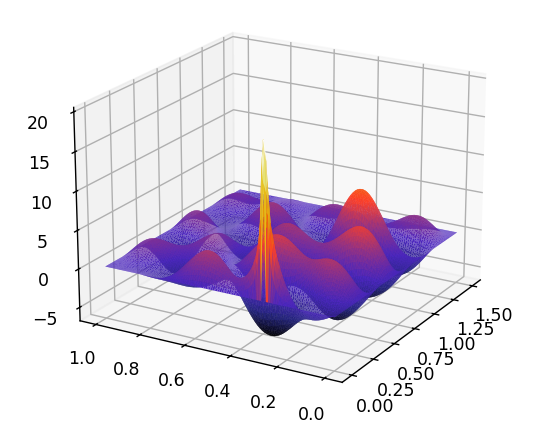


Рисунок 9 –

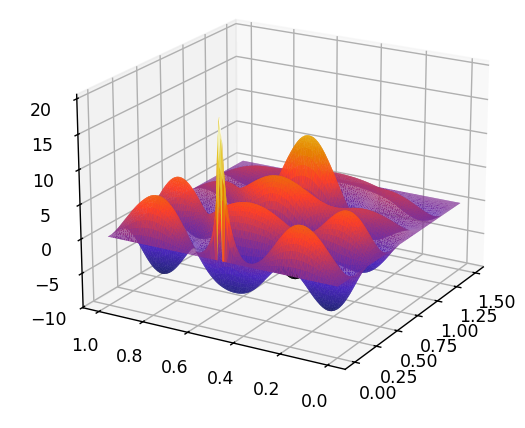


Рисунок 10 –

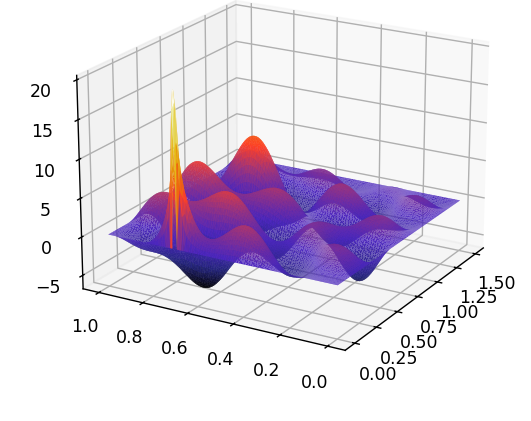


Рисунок 11 –

**Вывод**

В ходе лабораторной работы выполнение моделирование распространения электромагнитных волн в плоском неоднородном волноводе на основе решения краевой задачи для уравнения Гельмгольца. Написана вычислительная программа и проведены вычислительные эксперименты для выявляется влияния параметров на решение уравнения.

Значение отвечает за распространение возмущений по волноводу от левой границы до правой. От этого параметра зависит влияние функции на систему: при увеличении растет множитель при и уменьшается частота колебаний и амплитуда, так как неоднородность гасит колебания.

При изменении параметра было выявлено влияние на частоту колебаний волны. С увеличением параметра увеличивается и частота колебаний.

Параметр отвечает за смещение источника распространения волн относительно оси и на распределение волны. Это приводит так же к изменениям амплитуды.

**Приложение**

// matmod2\_3.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include<Eigen/Dense>

#include<Eigen/Core>

#include <Eigen/Sparse>

#include <unsupported/Eigen/IterativeSolvers>

#include <iostream>

#include <fstream>

#include <complex>

#include <vector>

#define Pi 3.1415926535

using namespace std;

int Nx = 400;

int Nl = 1000;

int Ny = 400;

double L = 1.0;

double Lpml = 0.5;

double k = 20;

const double ynull = 0.7;

double Fz(double y) {

double z = y;

if (abs(z) <= 2) {

cout << z << endl;

return 0.25 \* (1 + cos(Pi \* z / 2));

}

else return 0;

}

complex<double> gamma(double x) {

return 1.0+complex<double>(0, 1) / k \* pow((x - L) / Lpml, 2) \* 20.0 ;

}

complex<double> dgamma(double x) {

return complex<double>(0, 1) / k \* pow((x - L) / Lpml, 1) \*2.0 / Lpml\*20.0;

}

double fi(double y) {

return 1\*(0.5 - y) \* (0.5 - y) \* sin(Pi \* y);

}

int id(int i, int j) {

return i \* (Ny+1) + j;

}

int id(int i, int j, int l) {

return (Nx+1)\*(Ny+1)+2\*i \* (Ny+1) + j+l\*(Ny+1);

}

int main() {

double eps = 0.001;

double hy = 1.0 / Ny;

double hx = L / Nx;

Nl = int(Lpml / hx);

cout << Lpml << " b " << hx << endl;

int N = (Nx + 1) \* (Ny + 1) + (Ny + 1) \* (Nl + 1) \* 2;

Eigen::VectorXd f(N);

Eigen::SparseMatrix<double> s(N, N);

{

f.setZero();

//sys.setZero();

std::cout << "Hello World!\n" << Nx + Nl + 1 << " " << N << " " << (Nx + 1) \* (Ny + 1) << " " << (Ny + 1) \* (Nl + 1) \* 2 << endl;

std::vector<Eigen::Triplet<double> > triplets;

for (int i = 1; i < Nx; i++)

for (int j = 1; j < Ny; j++) {

int in = id(i, j);

triplets.push\_back(Eigen::Triplet<double>(in, id(i, j), (-2 / (hx \* hx) - 2 / (hy \* hy)) + k \* k \* (1 + eps \* fi(j \* hy))));

triplets.push\_back(Eigen::Triplet<double>(in, id(i - 1, j), 1 / (hx \* hx)));

triplets.push\_back(Eigen::Triplet<double>(in, id(i + 1, j), 1 / (hx \* hx)));

triplets.push\_back(Eigen::Triplet<double>(in, id(i, j - 1), 1 / (hy \* hy)));

triplets.push\_back(Eigen::Triplet<double>(in, id(i, j + 1), 1 / (hy \* hy)));

}

for (int j = 1; j < Ny; j++) {

f(id(0, j)) = Fz((j \* hy - ynull) / hy) / hy;

triplets.push\_back(Eigen::Triplet<double>(id(0, j), id(0, j), 1));

}

for (int i = 0; i < Nx + 1; i++) {

f(id(i, 0)) = 0;

triplets.push\_back(Eigen::Triplet<double>(id(i, 0), id(i, 0), 1));

f(id(i, Ny)) = 0;

triplets.push\_back(Eigen::Triplet<double>(id(i, Ny), id(i, Ny), 1));

}

for (int i = 1; i < Nl; i++)

for (int j = 1; j < Ny; j++) {

double x = L + i \* hx;

complex<double> c1 = 1.0 / (gamma(x) \* gamma(x));

complex<double> c2 = dgamma(x) / (gamma(x) \* gamma(x) \* gamma(x));

double a = c1.real();

double b = c1.imag();

double c = c2.real();

double d = c2.imag();

int id1 = id(i, j, 0);

int id2 = id(i, j, 1);

triplets.push\_back(Eigen::Triplet<double>(id1, id(i, j, 0), (-2 \* a / (hx \* hx) - 2 / (hy \* hy) + k \* k \* (1 + eps \* fi(j \* hy)))));

triplets.push\_back(Eigen::Triplet<double>(id1, id(i - 1, j, 0), (a / (hx \* hx) + c / (2 \* hx))));

triplets.push\_back(Eigen::Triplet<double>(id1, id(i + 1, j, 0), (a / (hx \* hx) - c / (2 \* hx))));

triplets.push\_back(Eigen::Triplet<double>(id1, id(i, j - 1, 0), (1 / (hy \* hy))));

triplets.push\_back(Eigen::Triplet<double>(id1, id(i, j + 1, 0), (1 / (hy \* hy))));

//////

triplets.push\_back(Eigen::Triplet<double>(id1, id(i, j, 1), (-2 \* -b / (hx \* hx))));

triplets.push\_back(Eigen::Triplet<double>(id1, id(i - 1, j, 1), (-b / (hx \* hx) - d / (2 \* hx))));

triplets.push\_back(Eigen::Triplet<double>(id1, id(i + 1, j, 1), (-b / (hx \* hx) + d / (2 \* hx))));

//////

if (i > 1) {

triplets.push\_back(Eigen::Triplet<double>(id2, id(i, j, 1), (-2 \* a / (hx \* hx) - 2 / (hy \* hy) + k \* k \* (1 + eps \* fi(j \* hy)))));

triplets.push\_back(Eigen::Triplet<double>(id2, id(i - 1, j, 1), (a / (hx \* hx) + c / (2 \* hx))));

triplets.push\_back(Eigen::Triplet<double>(id2, id(i + 1, j, 1), (a / (hx \* hx) - c / (2 \* hx))));

triplets.push\_back(Eigen::Triplet<double>(id2, id(i, j - 1, 1), (1 / (hy \* hy))));

triplets.push\_back(Eigen::Triplet<double>(id2, id(i, j + 1, 1), (1 / (hy \* hy))));

//////

triplets.push\_back(Eigen::Triplet<double>(id2, id(i, j, 0), (-2 \* b / (hx \* hx))));

triplets.push\_back(Eigen::Triplet<double>(id2, id(i - 1, j, 0), (b / (hx \* hx) + d / (2 \* hx))));

triplets.push\_back(Eigen::Triplet<double>(id2, id(i + 1, j, 0), (b / (hx \* hx) - d / (2 \* hx))));

}

}

for (int j = 1; j < Ny; j++) {

int id1 = id(Nx, j);

int id2 = id(0, j, 0);

triplets.push\_back(Eigen::Triplet<double>(id1, id1, 1));

triplets.push\_back(Eigen::Triplet<double>(id1, id2, -1));

/////////////////////////

triplets.push\_back(Eigen::Triplet<double>(id2, id1, 3));

triplets.push\_back(Eigen::Triplet<double>(id2, id(Nx - 1, j), -4));

triplets.push\_back(Eigen::Triplet<double>(id2, id(Nx - 2, j), 1));

//////////

triplets.push\_back(Eigen::Triplet<double>(id2, id2, 3));

triplets.push\_back(Eigen::Triplet<double>(id2, id(1, j, 0), -4));

triplets.push\_back(Eigen::Triplet<double>(id2, id(2, j, 0), 1));

///////////////////////////////

int id3 = id(0, j, 1);

triplets.push\_back(Eigen::Triplet<double>(id3, id3, 1));

//////////////////////////////

int id4 = id(1, j, 1);

triplets.push\_back(Eigen::Triplet<double>(id4, id3, -3 / (2 \* hx)));

triplets.push\_back(Eigen::Triplet<double>(id4, id4, 4 / (2 \* hx)));

triplets.push\_back(Eigen::Triplet<double>(id4, id(2, j, 1), -1 / (2 \* hx)));

//triplets.push\_back(Eigen::Triplet<double>(id4, id3, 1));

//triplets.push\_back(Eigen::Triplet<double>(id4, id4, -1));

}

for (int i = 0; i < Nl + 1; i++) {

f(id(i, 0, 0)) = 0;

triplets.push\_back(Eigen::Triplet<double>(id(i, 0, 0), id(i, 0, 0), 1));

f(id(i, Ny, 0)) = 0;

triplets.push\_back(Eigen::Triplet<double>(id(i, Ny, 0), id(i, Ny, 0), 1));

f(id(i, 0, 1)) = 0;

triplets.push\_back(Eigen::Triplet<double>(id(i, 0, 1), id(i, 0, 1), 1));

f(id(i, Ny, 1)) = 0;

triplets.push\_back(Eigen::Triplet<double>(id(i, Ny, 1), id(i, Ny, 1), 1));

}

for (int j = 0; j < Ny + 1; j++) {

triplets.push\_back(Eigen::Triplet<double>(id(Nl, j, 0), id(Nl, j, 0), 1));

triplets.push\_back(Eigen::Triplet<double>(id(Nl, j, 1), id(Nl, j, 1), 1));

}

s.setFromTriplets(triplets.begin(), triplets.end());

}

Eigen::SparseLU<Eigen::SparseMatrix<double> > solver(s);

//Eigen::GMRES<Eigen::SparseMatrix<double>> solver(s);

//solver.compute(s);

if (solver.info() != Eigen::Success) {

cout << "Failed" << endl;

}

Eigen::MatrixXd sol = solver.solve(f);

// cout <<f << endl;

// cout << s << endl;

// cout << sol << endl;

ofstream file("result" + to\_string(int(k)) +"y0"+ to\_string(ynull) + ".txt");

for (int j = 0; j < Ny + 1; j+=4) {

for (int i = 0; i < Nx+1; i+=4) {

file << j \* hy << " " << i \* hx << " " << sol(id(i, j)) << endl;

}

for (int i = 0; i < Nl + 1; i+=4) {

file << j \* hy << " " << L + i \* hx << " " << sol(id(i, j, 0)) << endl;

}

}

file.close();

int id1 = id(Nx, 49);

int id2 = id(0, 49, 0);

cout << 3 \* sol(id1) - 4 \* sol(id(Nx - 1, 49)) + sol(id(Nx - 2, 49)) << endl;;

cout << 3 \* sol(id2) - 4 \* sol(id(1, 49, 0)) + sol(id(2, 49, 0)) << endl;

//////////

return 0;

}